AFRL-RI-RS-TR-2010-127

XLAYER: A CROSS-LAYER COMMUNICATIONS SUBSTRATE FOR TACTICAL ENVIRONMENTS

Florida Institute for Human and Machine Cognition

June 2010

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■**UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2010-127 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/	/s/
KURT A TURCK	WARREN H. DEBANY, Jr.
Work Unit Manager	Technical Advisor, Information Grid Division
	Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* JUNE 2010	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* June 2007 – December 2009

4. TITLE AND SUBTITLE
XLAYER: A CROSS-LAYER COMMUNICATIONS SUBSTRATE FOR TACTICAL ENVIRONMENTS

5a. CONTRACT NUMBER N/A
5b. GRANT NUMBER FA8750-07-2-0185
5c. PROGRAM ELEMENT NUMBER 62702F

6. AUTHOR(S)
Marco Carvalho

5d. PROJECT NUMBER 558J
5e. TASK NUMBER CL
5f. WORK UNIT NUMBER CS

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Florida Institute for Human and Machine Cognition
40 South Alcaniz Street
Pensacola, FL 32502-6008

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/RIGC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2010-127

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The notion of a cross-layer communications substrate for tactical battlefield environments described in this work proposes to address the problem by enabling a two-way interface between higher-level applications, middleware or decision architectures, and the underlying communications infrastructure. The adaptation across multiple layers of the communication stack requires mechanisms capable of coordinating local adaptation at different time-scales to better allocate resources and capabilities in response to changes that may occur either on demand or proactively, based on explicit application requirement patterns. XLayer is a specialized cross-layer communications substrate designed to support a communications infrastructure for tactical and mobile ad hoc networks that provides the mechanisms to enable the interaction between applications and the different levels of the communications substrate to support a large range of scenarios and capabilities.

15. SUBJECT TERMS
Cross-layer Communication Substrate, mobile ad hoc networks (MANETs), Predictive Routing, Adaptive Discovery, Information Dissemination, Dynamic Gateway Selection

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Kurt A. Turck
U	U	U	UU	48	**19b. TELEPHONE NUMBER (*Include area code*)** N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.

ABSTRACT

Tactical and mobile ad hoc networks (MANETs) are key technology enablers for future mission-critical communication infrastructures such as those envisioned for disaster relief operations and military missions. While extremely flexible and powerful, these kinds of networks also create a new set of demands. The traditional layered approach that has shielded applications from the underlying networks is no longer applicable, and a new challenge has emerged—how to provide such interface or mutual awareness while avoiding customized stove-piped solutions.

The notion of a cross-layer communications substrate for tactical battlefield environments described in this work proposes to address the problem by enabling a two-way interface between higher-level applications, middleware or decision architectures, and the underlying communications infrastructure.

The adaptation across multiple layers of the communication stack requires mechanisms capable of coordinating local adaptation at different time-scales to better allocate resources and capabilities in response to changes that may occur either on demand or proactively, based on explicit application requirement patterns.

XLayer is a specialized cross-layer communications substrate designed to support a communications infrastructure for tactical and mobile ad hoc networks that provides the mechanisms to enable the interaction between applications and the different levels of the communications substrate to support a large range of scenarios and capabilities.

In this report we summarize some of the concepts, architectural design choices and capabilities of the XLayer communications substrate.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SUMMARY

Traditionally, cross-layer strategies for tactical and mobile ad hoc networks have primarily focused on short-term adaptation and state reporting between neighboring layers, generally for the optimization of a single protocol. Because local adaptations at each layer usually occur at different time-scales, a better coordination mechanism for an effective cross-layer communication infrastructure is required to allow applications and decision architecture systems to better adapt and meet Quality of Service (QoS) requirements and constraints.

We have developed XLayer, a cross-layer infrastructure for tactical networks that provides the necessary mechanisms to enable the interaction between applications or middleware and the different levels of the communications substrate, facilitating the integration of more complex functionality by leveraging the information and functionality provided by core services that abstract essential capabilities realized across the various layers of the communications stack.

XLayer supports several operating systems, multiple computing platforms, and tactical radios where specific capabilities enable seamless operation across multiple networks and communication environments in tactical systems and other applications in security and network management.

1 INTRODUCTION

In this report, we summarize our findings in the research for the development of XLayer, a cross-layer communications substrate designed to provide a modular communications infrastructure that allows applications to better adapt to the characteristics of dynamic communications environments, and support application requirements and constraints for tactical and mobile ad hoc networks.

Section 2 describes the methods, assumptions and procedures in the development of XLayer. Section 3 introduces the goals and capabilities of XLayer, summarizes the main characteristics of previous cross-layer approaches that addressed similar problems considered in our research, and gives a detailed description of the design and implementation of XLayer and its core components. Section 4 presents specific applications and discusses the results. Section 5 summarizes our conclusions and discusses the impact of XLayer in different projects and demonstrations. Finally, Section 6 presents our suggested course of actions.

2 METHODS, ASSUMPTIONS AND PROCEDURES

The XLayer communications substrate was designed to provide a modular framework for distributed monitoring and resource management in tactical environments. In this section we introduce our design assumptions and evaluation methods.

2.1 Design Assumptions

We consider a mobile wireless network environment where each platform may potentially have multiple communications interface, connected through shared or point-to-point network links. As illustrated in Figure 1a**,** our target scenario includes a conventional link-uniform topology used in MANET research, where platforms are assumed to communicate through a common network medium that is shared between all nodes. In Figure 1b, the true underlying topology is shown, with connections between the different interfaces.

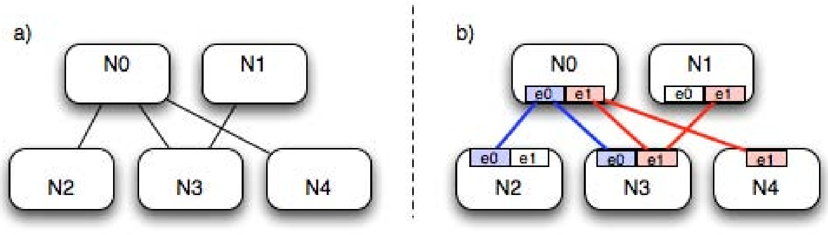


Figure 1. Overlapping Topologies

2.1.1 Visibility

Each interface can be fully monitored and attributes at different layers can be configured, at run time, by the proposed infrastructure. Visibility is an important capability for resource management and coordination across different nodes and applications. Low-level network visibility can be achieved through monitoring APIs provided by the Operating System, radio-specific APIs, or customized components that communicate with the radio interface.

Monitoring and control of network and transport protocols require some level of code change. Alternatively, low-level data packets, such as those at the MAC-level, can be mapped and correlated to routing and transport, if not encrypted.

Application-level monitoring also requires some level of collaboration or feedback from applications, which may be provided, for instance, through common APIs used by applications to monitor and report system resources, protocols, and services.

2.1.2 Control

Also as part of this work, we assume that configuration and control APIs for the system are available at run-time. Due to time-scale requirements of different layers and communication components, it is important that both monitoring and control APIs can operate in different time scales, and yet share the necessary messages and information for coordination.

We assume that each node has full control over its communications and computational resources, and can reconfigure its settings at all layers, at run time. Such configurations include, for instance, physical network interface settings, beaconing parameters and window sizes of higher-level protocols, and also application-level restrictions for resource utilization, computational load, data shaping, etc.

2.1.3 Distributed Coordination

Lastly, we assume that nodes can share state information and coordination messages efficient for distributed network management. The sharing of information must be done efficiently and must minimize the co-dependency between services and applications. We assume that high-level services and applications can be modified to utilize messaging and control APIs provided by the infrastructure for discovery and information exchange.

2.2 Project Goals and Requirements

The primary goal of the proposed project was to create a communication substrate (or middleware) that would closely interface low level protocols and capabilities in the tactical network environment with high level distributed applications and information management systems. The goal was to establish this seamless cross-layer interface to improve the agility, visibility and coordination of information management systems, while maintaining a loose coupling between the applications and communications infrastructure.

As a proof-of-concept of the proposed cross-layer capabilities, we have scoped our project to address four areas of special interested, while focusing on the architectural design, core capability development, modularity and extensibility of the proposed infrastructure. The four areas chosen as proof-of-concept for our developments are described in the following paragraphs.

2.2.1 Seamless Support of Heterogeneous Networks

Actual battlefield deployments generally include a number of tactical radios with different link and data rate capabilities. The technical characteristics of different interfaces and data links may vary significantly in terms of capacity, latency, medium access policies, etc.

Links are connected to an interface and applications bind to specific interfaces to communicate with one another, effectively forming sets of overlapping networks that are connected through some of the multi-interface nodes in the database. A capability of interest for the proposed substrate is to seamlessly communicate across the different networks, allowing applications to locate, and address each other effortlessly, taking advantage of the best communication paths between the two end-points.

Such capability should be achieved transparently, not requiring applications to maintain separate links to multiple interfaces, or even being aware of multiple interfaces and networks.

The XLayer communications substrate achieves this capability by creating a common network service that controls each interface independently (see 3.3.2.7). This capability was also demonstrated in simulation on an emulated environment using different links and tactical radios (see 4.7).

2.2.2 Adaptive Discovery and Information Dissemination

In the context of this work, discovery refers to the process through which a node becomes aware of other nodes and the services they provide. Nodes make use of the discovery mechanism to register services and advertise the availability of computational resources through the dissemination of messages across the network. Likewise, the dissemination of queries enables nodes to find other nodes and services with specific capabilities and resource availability.

Because of the lack of a fixed infrastructure and the presence of nodes that are allowed to move freely through the network, discovery and information dissemination is often accomplished in MANETs by the broadcasting of packets. The most simplistic form of broadcasting, called *flooding*, typically causes unproductive and harmful bandwidth congestion as each node retransmits each received packet exactly once.

Many researchers have proposed more efficient broadcasting techniques whose goal is to minimize the number of retransmissions and thus reduce the overhead of information dissemination in MANETs. Although more efficient than simple flooding, the effectiveness of these techniques to reduce congestion and resource utilization greatly depends on the network topology and traffic conditions [20].

We have designed a hybrid discovery and information dissemination mechanism that is capable of self-adapting to different network topologies and traffic scenarios. This self-adapting mechanism monitors the network at different levels to make use of a broadcasting algorithm that is more suitable for a set of localized network conditions. The evaluation results of our approach are discussed in section 4.4.

2.2.3 Improved Routing and Transport Support

Information gathered from physical and network layers can also enable the improvement of routing and transport protocols for enhanced application or information management system support.

At the routing level, we have investigated a number of cross-layer strategies for improvement, including specialized multicast implementations [10][11], and predictive routing protocols, detailed in section 4.1.

We have also explored these capabilities for cross-domain routing, designing a dynamic gateway selection controller that seamlessly, and on demand, created proxy nodes to bridge across heterogeneous routing protocols [7]. Dynamic gateway selection is further discussed in section 4.5.

In terms of transport protocols, we have investigated dual-link strategies to improve the reliability or throughput of transport protocols, in response to requirement defined by higher-level services and applications.

2.2.4 Improved Visibility of Network State

Network visibility is a critical capability for adaptive information systems and applications. Critical information includes the configuration of computational platforms and interfaces, as well as the demand (e.g. ongoing data flows and computational requirements) and availability of resources.

The XLayer communications substrate provides the bases for a distributed monitoring infrastructure that enables both the collection, aggregation and sharing of node state information for resource management. The XLayer distributed monitoring service was applied in support of a related AFRL project entitled QoS-Enabled Data Dissemination [13][14][15], and is further discussed in section 4.6.

Information disseminated through the Monitoring Service is available to other services and applications at each node, and used for tasks that include dynamic resource allocation and planning.

2.3 *Design Approach and Evaluation Procedures*

The XLayer substrate is, by design, a collection of control algorithms that share a common representation, data-storage, communications, and coordination infrastructure. As detailed in Section 3, algorithms implemented as XLayer controllers can be instantiate and parameterized on demand to support application requirements or other services. Our experimental validation and evaluation procedures varied for different algorithms and contexts, but they were generally based on numerical simulations, emulation experiments in controlled environments, and field test exercises, which were primarily used for demonstrations and qualitative results.

2.3.1 Experimental Studies and Evaluations

In the early stages of the project, we relied on the NS-2 simulator for testing and evaluation, which was later replaced with NS-3 as our primarily simulation environment.

In addition to simulation, we have also developed an emulated network environment for development and evaluation where different link conditions can be quickly emulated to recreate a dynamic network topology, while allowing the XLayer substrate, information management system and applications to run in virtual machines operating at standard operating system time-scales. Emulation environments are very important to support the evaluation of systems and protocols that are time-sensitive, such as routing and reliable transport protocols like TCP. For such studies, the event driven nature of simulated environments often create artifacts on the data that may compromise the results. In support to this task we have developed a testbed emulation environment that is described in section 3.5.

In addition to emulated environments for development and experimentation, we have also implemented the cross-layer substrate for actual field test and evaluation. Figure 2 illustrates some of the computational platforms and tactical radios where we tested the XLayer substrate.

Serial and tactical radios such as the Mircrohard Spectra, the PCS-5D, and the EPLRS/Microlights were supported through their serial or IP interfaces. For those radios, the XLayer substrate was running at the computer to which the radios where connected, and managed the radios through a virtual interface in the Network Management Service (see 3.3.2.7).

For other devices with accessible computational platforms, such as the Lynksys base-stations, the Soekris boards, iPhone and iPod Touch platforms, and others, the XLayer was ported to the native operating system to directly manage the communications interfaces.

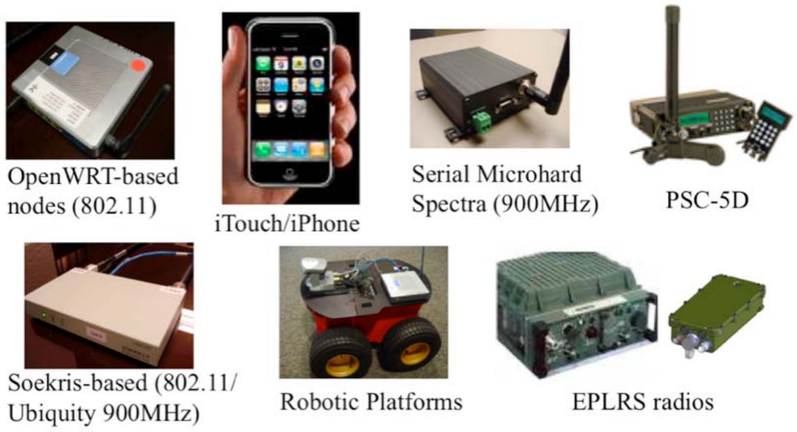

OpenWRT-based nodes (802.11)
iTouch/iPhone
Serial Microhard Spectra (900MHz)
PSC-5D
Soekris-based (802.11/ Ubiquity 900MHz)
Robotic Platforms
EPLRS radios

Figure 2. Some of the Platforms and Tactical Radios Tested with the XLayer Substrate

The operating systems supported for the Linksys radios were the OpenWRT (both the White Russian and Kamikaze versions). In those platforms, the XLayer managed all the wired and wireless interfaces, and also controlled the USB interfaces to enable GPS and external storage capabilities for logging.

The Soekris boards (http://www.soekris.com/) supported multiple network interfaces through MinPCI cards using the Pyramid operating system. Both the OpenWRT and Pyramid OS implementations of the XLayer provided full monitoring of the communications interfaces, and computational resources such as memory and CPU utilization. Limited control of physical layer properties (such as transmission power) was also available on some of the platforms.

3 THE XLAYER COMMUNICATIONS SUBSTRATE

3.1 Introduction

XLayer was designed to provide a modular communications infrastructure that allows applications and decision architecture systems to better adapt to and leverage the characteristics of the dynamic communications environment. It also enables the underlying communications infrastructure to better support application requirements and constraints.

In support of the application and decision architecture systems, XLayer monitors, abstracts, and represents the characteristics and capabilities of the underlying communications infrastructure so applications can better adapt to changes in the underlying communications environment (e.g., by re-allocating resources).

In support of the communications infrastructure, applications can provide information about resource requirements (both computational and communications) or utilization patterns. This information can then be used by the underlying communications infrastructure to better allocate resources and capabilities in response to changes that may occur either on demand or proactively, based on explicit application requirement patterns.

3.2 Related Work

Traditionally, cross-layer strategies for tactical and mobile ad hoc networks have primarily focused on short-term adaptation and state reporting between neighboring protocol layers, generally for the optimization of a single protocol such as Transmission Control Protocol (TCP) [12].

In general, most implementations are based on variations of QoS protocols inherited from the wired networks and still utilize some of the notions of signaling and coordination of neighboring protocol layers for resource reservation. The goal of most traditional cross-layer strategies is to monitor and detect short-term changes in channel conditions or competing traffic to notify upper layers about new QoS conditions. In most cases, applications are generally expected to adjust data rates accordingly when notified by a neighboring layer that current service expectations are no longer available.

As illustrated by Goldsmith and Wicker [9], the actual adaptation and reporting between layers is generally done after local layer adaptations are no longer possible or cost effective. The different time-scales at each layer usually imply that local adaptation within each layer generally occurs first, and more frequently, than adaptation between layers.

Protocols like dRSVP [16], for instance, provide per-flow end-to-end bandwidth guarantees for a range of requirements as opposed to a specific requirement like in RSVP. In this case, dRSVP "routers" exchange bandwidth reservation details through a signaling protocol and the flow is either denied access or dropped if channel availability becomes insufficient.

The SWAN Protocol [2] also uses signaling for short-term resource reservation. SWAN, like dRSVP, is fully decentralized, but it is best effort only and makes no assumptions about underlying QoS capabilities from the Medium Access Control (MAC) layer. The signaling in SWAN is intended for flow admission and the cross-layer nature of the protocol lies in the fact that MAC level packet delay information is shared and used for estimating medium access contention. After a flow is admitted in SWAN, the protocol uses the packet's explicit congestion notification flag (ECN) to notify that requested services are no longer supported for that flow.

TIMELY [3] is another cross-layer architecture that provides link layer scheduling, resource reservation and adaptation, as well as priority-aware transport protocol that self-regulates flow based on feedback from the lower layers. TIMELY was initially proposed for cell-based wireless networks, and helped create the basis for subsequent ad hoc specific architectures and protocols with similar capabilities like Spine [18] and CEDAR [17].

3.3 The XLayer Architecture

XLayer is specifically designed to provide and support a communication infrastructure that allows applications and decision architecture systems to better adapt and meet QoS requirements and constraints. A modular architecture makes XLayer flexible enough to support different scenarios that require of very specific capabilities.

In the XLayer architecture (Figure 3), each service module provides a set of capabilities that can be directly utilized by other models, by loadable sub-controllers, or even by overlaying applications. Furthermore, state information created or maintained by different modules can be registered and made available to other components within the architecture. The goal is to facilitate the sharing of state information, avoiding the redundant rediscovery costs and overhead. For example, neighborhood and link detection are critical tasks for several types of applications and services (e.g. routing, discovery, data dissemination, etc.). XLayer enables neighborhood and link information to be easily shared to avoid the existence of redundant detection mechanisms on each of these services.

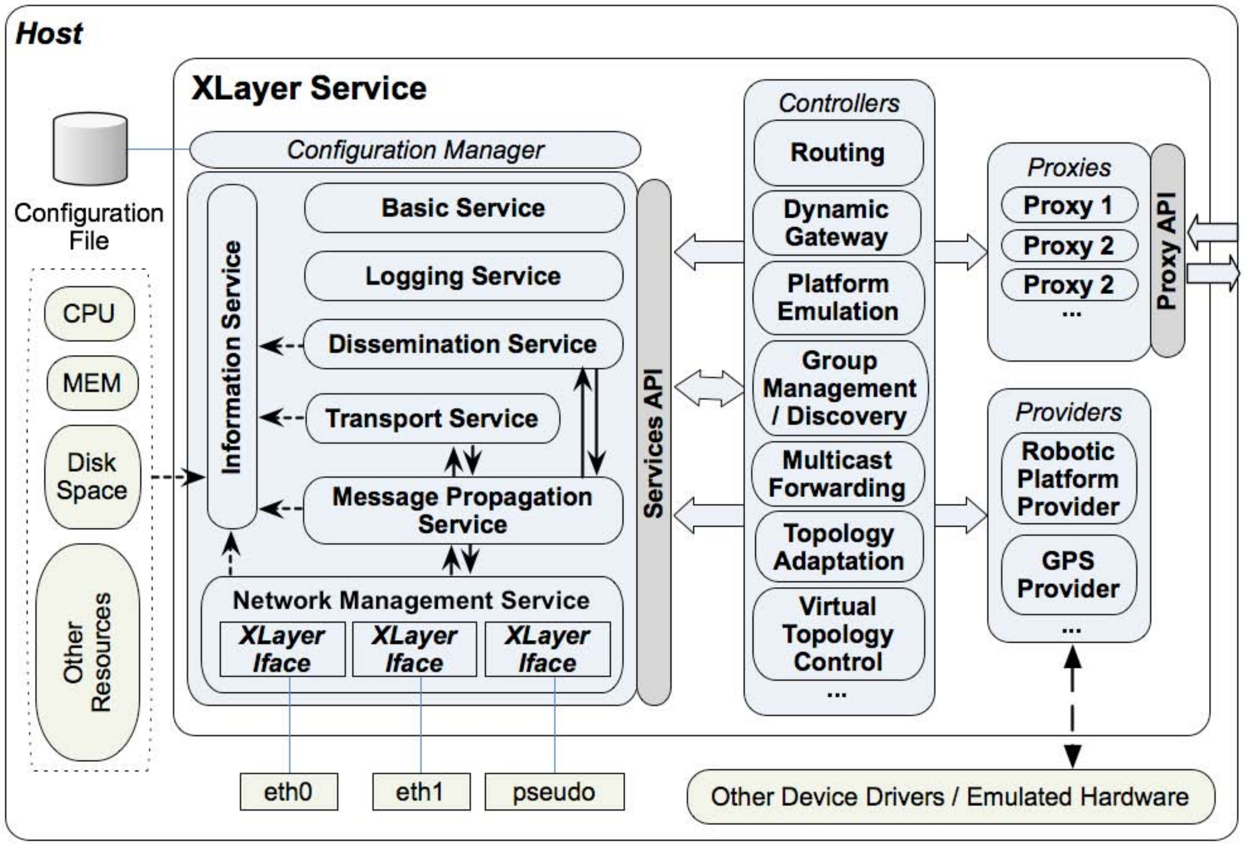


Figure 3. The XLayer Architecture

XLayer supports two types of modules: services and controllers. XLayer services provide essential capabilities to other XLayer components and are typically started upon instantiation of the XLayer. XLayer controllers allow developers to extend the functionality of the XLayer and can be enabled or disabled depending on the requirements of client applications.

In general, client applications communicate with the XLayer service using TCP sockets and a proprietary binary protocol. In order to simplify the development of these client applications, a proxy library, which is available in C++ and Java, enables remote procedure calls by handling the marshaling and unmarshaling of function parameters and results.

The proxy handles the registration of several callback mechanisms. These callbacks inform client applications of the state of the channel between the application and the XLayer service (connected, disconnected), reception of messages, and monitoring events from metric and property updates. Additionally, the proxy provides a mechanism to create datagram and stream-oriented sockets that may be used transparently by applications to take advantage of the XLayer transport capabilities such as dual-path flows, multi-path routing, adaptive transport, and reliable UDP-based transport [4].

3.3.1 The XLayer Proxy API

The XLayer Proxy API is implemented in C++, but a Java Native Interface (JNI) wrapper allows Java applications to interact with the XLayer as well. The XLayer Proxy enables applications to transparently take advantage of the capabilities of XLayer based on a rich set of APIs that core services and controllers implement and expose through well-defined mechanisms. Existing native and Java applications may, for example, use datagram and stream-oriented sockets that transparently interface with the XLayer transport service to offer capabilities such as adaptive and dual-path transport and multi-path routing.

The C++ and Java XLayer proxy implementations provide the necessary APIs to connect and communicate with an XLayer service running locally or in a remote host. The following code fragment in C++ shows how to create an instance of the proxy to connect to an XLayer service running locally:

```
#include “XLayerProxy.h”
...
XLayerProxy *pProxy = new XLayerProxy();
pProxy->connect();
...
```

In the previous C++ example, the XLayerProxy will attempt to connect to the XLayer service running on the same host. That is, the XLayerProxy will try to open a connection to *localhost* on port *2000*. The invocation of the connect method will not return until the XLayerProxy manages to establish a connection with XLayer. For this reason, in case the XLayerProxy needs to be connected to a XLayer service running on a remote host or in a port different (other than the default one), a connection can be made as follows:

```
#include “XLayerProxy.h”
...
XLayerProxy *pProxy =  new XLayerProxy();
pProxy->connect("192.168.1.1", 2000);
...
```

Note that the call to the *connect ()* method may block if not XLayer is running on the given address and port number. In this case, a timeout value in milliseconds (2000 milliseconds, for example) can be specified as follows:

```
#include “XLayerProxy.h”

...
XLayerProxy *pProxy = new XLayerProxy();
pProxy->connect(“192.168.1.1”, 2000, false, 3000);
...
```

The proxy can also connect to XLayer asynchronously. In this case, the proxy notifies the client application through the *XLayerProxyListener* interface when the connection to the XLayer service is established. In the same manner, the proxy notifies the client application when it is disconnected from XLayer.

```
#ifndef XLAYERPROXYLISTENER_H_
#define XLAYERPROXYLISTENER_H_

class XLayerProxyListener
{
      public:
            virtual ~XLayerProxyListener();
            virtual void connected() = 0;
            virtual void disconnected() = 0;
};

#endif /* XLAYERPROXYLISTENER_H_ */
```

Once connected, the application can use the capabilities of XLayer through the XLayer Proxy API.

3.3.2 XLayer Services

XLayer services are modules that provide the essential capabilities required by other components to perform more complex tasks (See Table 1). In the XLayer, a set of core services provide support for network interface detection and configuration, transport of messages, flooding, neighbor discovery, and monitoring of the different node and network properties, such as resource utilization, link quality, topology, and route information.

Table 1. XLayer Services

Service Name	Main Capabilities
Basic Service	Provides bootstrap APIs for XLayer's controllers and services.
Logging Service	Provides a mechanism for remote logging of the XLayer service.
Information Service	Constitutes the central repository of shared information in XLayer.
Dissemination Service	Provides a common API for a configurable set of message flooding mechanisms for MANETs.
Transport Service	Provides end-to-end communication for XLayer-enabled nodes.
Message Propagation Service	Provides 1-hop communication with other XLayer-enabled nodes.
Network Management Service	Detects and manages all physical (wired and wireless) network interfaces that are available to XLayer.

3.3.2.1 Basic Service

The Basic Service makes available two key functionalities to other XLayer modules: 1) a mechanism to load/start and unload/stop XLayer controllers, and 2) a mechanism to add or remove bridge connections. A bridge connection is a permanent TCP connection that is established with another XLayer service to interconnect two or more XLayer-aware networks with different address spaces, transport protocols, or routing algorithms.

3.3.2.2 Logging Service

The Logging Service provides a mechanism for remote logging of the XLayer service. Log messages are intercepted and sent over a UDP socket to facilitate monitoring and debugging of XLayer modules executed on platforms which can only be accessed remotely or have very limited storage space.

3.3.2.3 Information Service

The Information Service constitutes the central repository of shared information in XLayer. It holds and provides access to information about the local and remote nodes, as well as the existing links between them. The Information Service uses the available information to compute routes and link quality metrics such as packet drop rates and delays, and advertises the properties of the local node such as number of interfaces, CPU usage, memory, disk and network utilization. Additionally, it provides synchronous and asynchronous notification mechanisms for node discovery and status and topology changes.

3.3.2.4 Dissemination Service

The Dissemination Service provides a common API for a configurable set of message flooding mechanisms for MANETs. Flooding is often used by numerous routing and distributed coordination algorithms in MANETs, but it generally constitutes a very expensive operation in mobile ad hoc networks [20]. The Dissemination Service enables applications to use the most suitable flooding algorithm given the current network conditions. Additionally, it provides a mechanism for registering messages that need to be repeatedly and periodically disseminated at specific time intervals, effectively enabling a powerful API for proactive message propagation in the network.

3.3.2.5 Transport Service

The Transport Service provides end-to-end communication for XLayer-enabled nodes and the ability to track and split flows in order to support dual-path algorithms. The reception and forwarding of messages is done through the Message Propagation Service, however, the Transport Service is responsible for determining which interface(s) will be used to forward the message. This makes possible for the Transport Service to provide optimized, reliable and adaptive transport capabilities to accommodate the QoS requirements of other components and applications.

3.3.2.6 Message Propagation Service

The Message Propagation Service provides 1-hop communication with other XLayer-enabled nodes through unicast, broadcast or multicast UDP packets that encapsulate one or more XLayer messages. XLayer controllers, services and applications can register message handlers for specific or new message types. In addition, the Message Propagation Service provides basic neighbor discovery and link sensing, consolidated transmission of multiple messages to reduce bandwidth utilization, and extensible packet headers and packet filters that can be used to enable topology emulation capabilities, instrumentation, monitoring and other important features required by controllers and middleware applications.

3.3.2.7 Network Management Service

The Network Management Service detects and manages all physical (wired and wireless) network interfaces that are available to XLayer. Additionally, it gathers statistics on the number of packets (and bytes) transmitted and received on each of the network interfaces. It also makes available an API to activate and deactivate interfaces, and to notify other modules about their status (up, down).

3.3.3 XLayer Controllers

The functionality of the XLayer can be extended through XLayer controllers. An XLayer controller is a module that implements one or more algorithms to perform specific tasks (See Table 2). In many cases, these algorithms can be made available to other XLayer modules and composed to further extend the XLayer functionality.

Table 2. XLayer Controllers

Task	Description
Group Management and Discovery	A set of controllers provides basic group management and discovery. These controllers use the Dissemination Service to permanently advertise group membership and perform peer searches within a certain scope (i.e., hop-distance).
Adaptive and Predictive Routing	A predictive routing controller is responsible for correcting the offset of *hello* messages of a link-state routing protocol to construct a projection of the topology for next hop selection. In our previous research [8], we have shown significant improvements in both packet loss and average delay for predictive routing by simply adjusting the local topology based on short-term mobility and link quality trends for neighbor nodes.
Dynamic Gateway Selection	A controller allows XLayer to use different routing strategies that can help to reduce routing related traffic on ultra dense networks.
Multicast Forwarding	A controller provides efficient multicast packet forwarding in MANETs based on the approach followed by the OLSR Basic Multicast Forwarding (BMF) plug-in [19] and the capabilities of the Dissemination Service.
Topology Adaptation and Control	A set of controllers makes use of information about the flows of data going through the neighbors of the local node to change its position based on the application's transport requirements.
Virtual Topology Control	A controller registers an extended header and filter to inspect all packets received by the Message Propagation Service in order to accept or reject packets based on the sender and receiver node's virtual position and their transmission power, constructing a virtual topology that can be used to setup simulations and experiments.
Platform Emulation	A set of controllers enables applications to transparently access node's properties such as position, direction, and speed, without making any assumptions about the environment where XLayer is running.

XLayer can dynamically load XLayer controllers. The dynamic loading of XLayer controllers has three main advantages: 1) it makes possible to extend the functionality of the XLayer without changing the XLayer executable, 2) it encourages the reutilization of code by sharing common functionality among controllers, and 3) it facilitates the integration XLayer controllers by third-parties.

3.4 Supported Platforms and Proof-of-Concept Implementation

As described in the previous section, a proof-of-concept implementation of the XLayer architecture was developed in C++, with proxy APIs for C++ and Java. This implementation was extensively tested and demonstrated in different environments, and has been used to evaluate some of the controllers described in Section 3.3.3.

3.5 Emulation Environment based on XLayer

Test and validation of MANET technologies and applications have always been a complex and challenging problem. Theoretical models of such networks have traditionally been used as the basis for simulation and emulation frameworks. However, in most cases, such models are generally too complex to be practical or too simplistic to be representative.

Not surprisingly, the alternative approach for validation and development is to rely on field experiments which are usually costly, time consuming and very difficult to replicate. In the case of airborne networks, the issue is even more complicate, as it involves more degrees of freedom for the participating nodes and multiple external effects such as shadowing caused by the airframe, engine interference and others. All these factors are very difficult to model and represent in a simulated or emulated environment accurately.

As an extension to this project, we executed a task to design and develop a hybrid emulation testbed for mobile ad hoc networks called MLAB. MLAB enables both the use of theoretical propagation models and experimental data to emulate link characteristics between nodes. Furthermore, it allows for a mixed modeling strategy that includes both the theoretical models and experimental field data.

These characteristics make it a well-fit emulation environment for airborne networks, where the multiple degrees of freedom and complexity of the nodes make it very difficult to create reliable theoretical models that would suffice for the emulation.

3.5.1 The MLAB Testbed

Figure 4 illustrates the basic architecture and the physical layout of the MLAB testbed. Note that each testbed node has two interfaces, one connected to the control network, and the other connected to the data network. The only exception is the controller machine, which is only connected to the control network because it plays no role on data exchange. The design physically separates control from data traffic, minimizing the effects of monitoring and control in the actual experiment.

The first version of the emulation environment made use of a Linux service for tagging all IP packets transmitted through the data interface and a kernel library for capturing and filtering those packets in order to emulate the required link conditions. The second implementation of the MLAB testbed used a virtual network interface for link enforcement based on the TUN/TAP virtual network kernel driver, providing a much more flexible and faster mechanism for link parameterization.

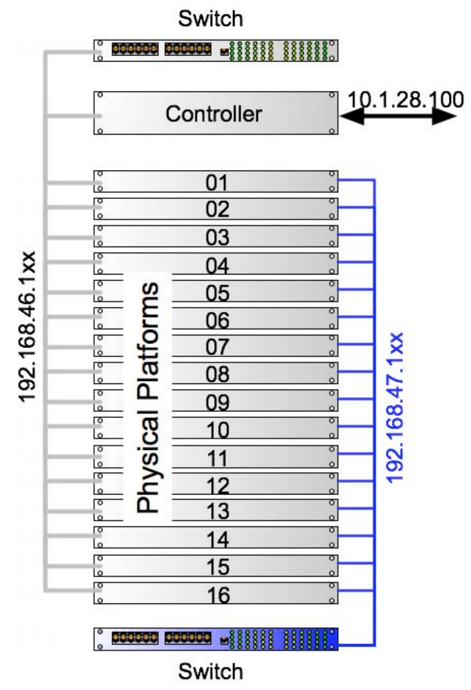


Figure 4. The MLAB testbed: the control (gray) and data (blue) networks

Parameters for the TUN/TAP virtual interface are provided at run-time by a link modeling application running at the controller node (see Figure 5), for a given network topology and mobility scenario. All data coordination messages between the controller and enforcement drivers are done through an isolated network known as the Control Bus. On the other hand, all data messages between applications running on the emulation network are exchange through the Data Bus, which is separated from the Control Bus to minimize any coordination overhead that may affect the actual emulation.

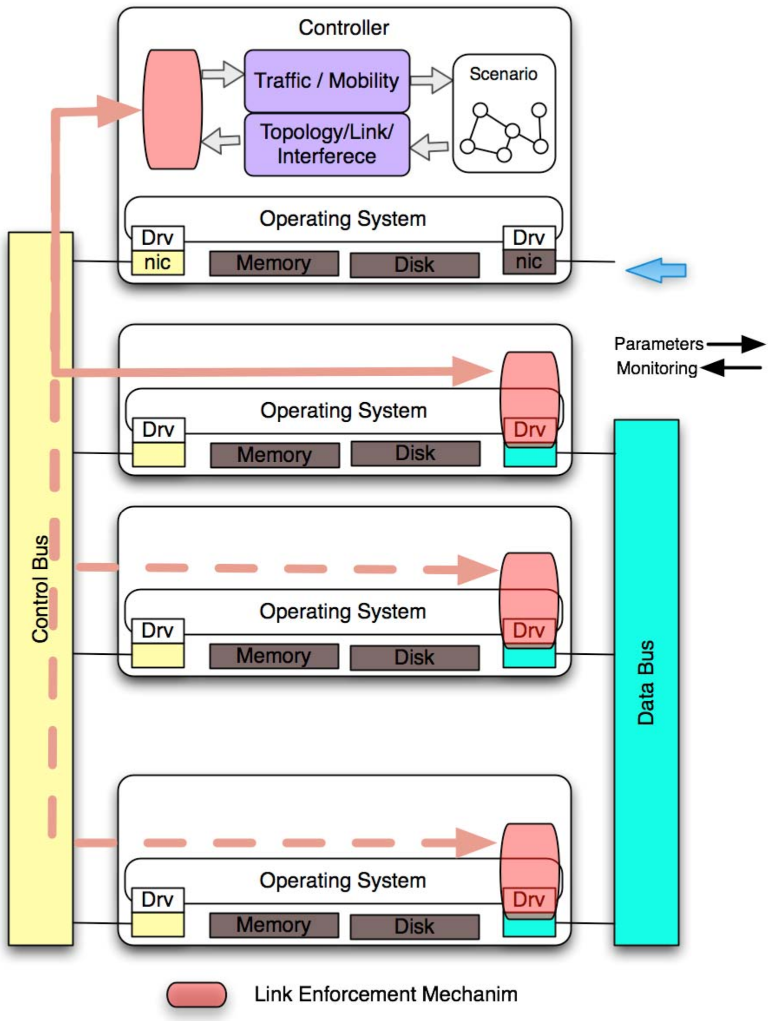


Figure 5. Main Coordination and Enforcement Components of the MLAB testbed

3.5.2 XLayer and MLAB Integration

XLayer enhances the usability of MLAB by providing a two-way interface between higher-level applications and the emulation components (Table 3). At each node, XLayer provides monitoring and control capabilities that enable users to transparently interface their applications with the underlying emulation and communications infrastructure. Applications use a proxy to communicate with the XLayer service to gather environmental information and to control transmission power and node mobility.

Table 3. MLAB Software Components

Name	Description
Controller Daemon	Manages and controls the emulation environment. This daemon maintains a global view of the network topology for each active experiment and monitors the attributes of each node (position, speed, and transmission power) to continuously adjust the link characteristics based on the packet drop and delay probability values indicated by the theoretical and data-driven models.
Node Daemon	Creates and manages the node's virtual emulation network interface. It continuously reconfigures the interface to filter incoming and outgoing packets according to the packet drop and delay probability values as indicated by the controller. This daemon also collects traffic statistics and periodically sends feedback messages to the controller to be used for calibration of the interference model.
XLayer	Provides monitoring and control capabilities that enable users to transparently interface their applications with the underlying emulation and communications infrastructure.
MView	Acts as a visualization and management tool for experiments. It allows users to create, edit and remove experiments as well as to visualize the network topology in a 3-D world. It also provides a tool for setting the attributes of each node such as speed, transmission power, position, and others.

When XLayer is used to interface applications with MLAB, the changes made at the cross-layer level are relayed to the controller and the link characteristics are adjusted to reflect the new conditions (Figure 6). Hence, the use of XLayer allows for easy testing of applications because no assumptions need to be made about the underlying platform, facilitating the deployment task outside the emulation environment.

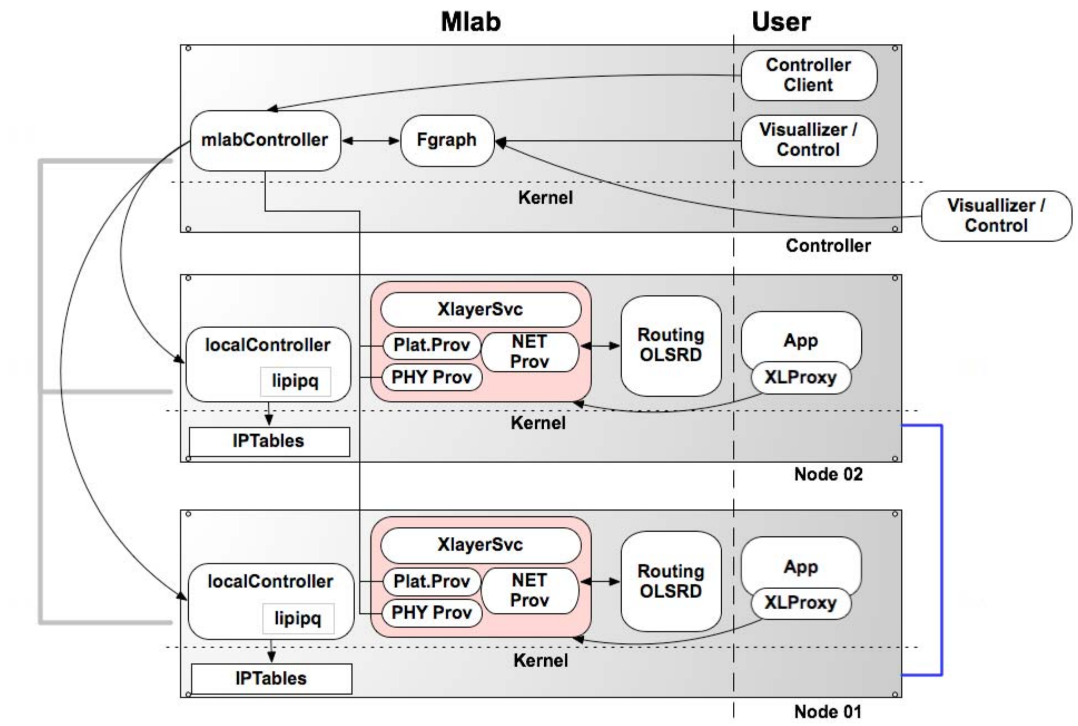


Figure 6. Cross-Layer Services for the MLAB Emulation Environment

4 RESULTS AND DISCUSSION

During the life of the project, the XLayer was deployed, tested and evaluated in different scenarios and demonstrations. In general each of the controller capabilities were tested independently, in simulation, emulation of field-test environments. In this section we provide a brief description of some of these studies, and refer the reader to additional publications for further reading.

Each of the capabilities described in this section were development as XLayer controllers and tested in one or more of the test platforms described in Section 3.4. Collectively, they realize the requirements specified in Section 2.2, and constitute the results of the proposed research effort.

4.1 Predictive Routing

In collaboration with the Army Research Laboratory, we have levered the distributed monitoring capability (see 4.6) of the XLayer substrate to create POLSR, a predictive routing protocol based on OLSR.

In most link state-based protocols like OLSR, changes in local topology are often detected through periodic beaconing. Because of the distributed nature of the protocol, each node is responsible for their own detection and reporting of unidirectional changes – which are then aggregated and shared by other nodes upon notification of change. While efficient, the approach leads to a slight inconsistency of state information that could become relevant for some scenarios. This effect is illustrated in Figure 7, where a snapshot is shown for a mobile network at four different times from left to right. In the figure, solid nodes have a correct representation of the correct routing table, while open nodes have inaccurate information induced by the changes in the network. As time progresses, the number of nodes with a misrepresented view of the global topology grows.

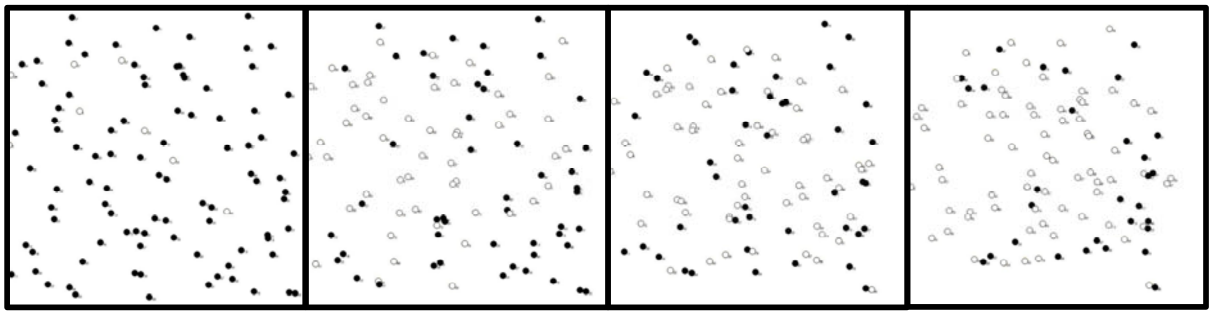
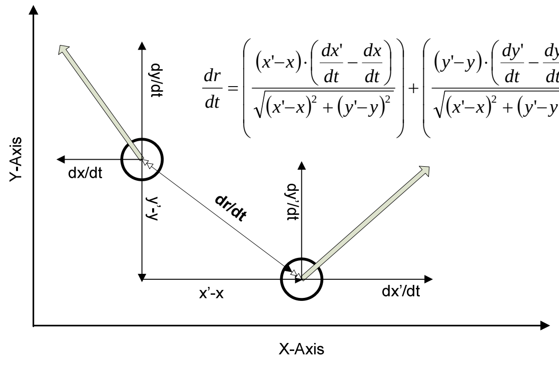


Figure 7. Mobility-induced routing table deviations with time (from left to right)

POLSR mitigates the inconsistency of state information by enabling each node to estimate, based on the mobility of its neighbors, the changes in the network to improve route calculation. The estimation of node mobility is indirect – either based on variations of signal strength capture from lower layers, or based on mobility vectors explicitly reported by nodes as part of their beaconing messages.

In this paper, we show that a local correction of topology based on simple mobility and propagation models. As shown in Figure 8, a distributed corrective model can be added to OLSR for route calculation. In Figure 8, P_t is the transmit power, and P_r the received power. G_t and G_r are the antenna gains for the transmitter and the receiver, L is the system loss, and λ is the wavelength. Based on the parameterized distance (d) between transmitter and receiver the received power function ($P_r(d)$) can be used to estimate the average and standard deviation of the log of the power at the receiver node $\log(P_r(d))$.

Each node still makes an independent estimation of network topology for calculating the routing table but it takes the beaconing delays, and the projected mobility patterns of its neighbors into account.



$$\frac{dr}{dt} = \left(\frac{(x'-x)\cdot\left(\frac{dx'}{dt}-\frac{dx}{dt}\right)}{\sqrt{(x'-x)^2+(y'-y)^2}}\right) + \left(\frac{(y'-y)\cdot\left(\frac{dy'}{dt}-\frac{dy}{dt}\right)}{\sqrt{(x'-x)^2+(y'-y)^2}}\right)$$

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d_0^2 L} \cdot \left(\frac{d}{d_0}\right)^{-\beta} \cdot 10^{X_{dB}/10}$$

$$\mu = \log\left(\frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d_0^2 L}\right) - \beta \log\left(\frac{d}{d_0}\right)$$

$$\sigma^2 = \left(\frac{\sigma_{dB}}{10}\right)^2$$

Figure 8. Mobility and Propagation Models for POLSR

As described in [7], the POLSR enabled significant gains in reduced packed loss (Figure 9) and jitter for random walk scenarios over uniformly distributed network topologies.

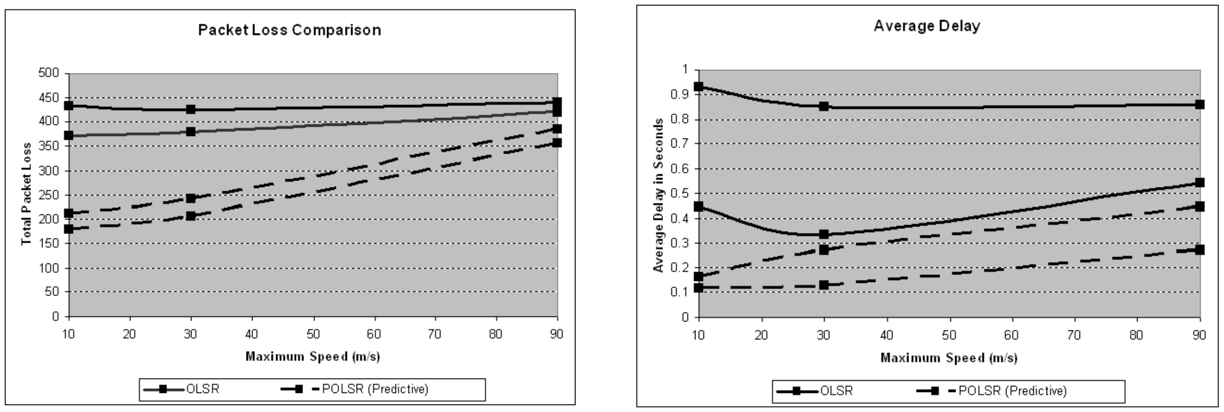


Figure 9. POLSR Experimental Results

4.2 *Adaptive Dual-Link Allocation for Transport Support*

One of the capabilities provided by the XLayer substrate was the abstraction and transparent utilization of multiple links (or overlapping networks) on each node. Data links with different characteristics and capabilities could be used, in different contexts, to support information management system or application level requirements.

On the example scenario illustrate in Figure 10, a pub-sub configuration was created using the AFRL's reference implementation of the Apollo Information Management System (later replaced by the AFRL's tactical IMS implementation called Phoenix).

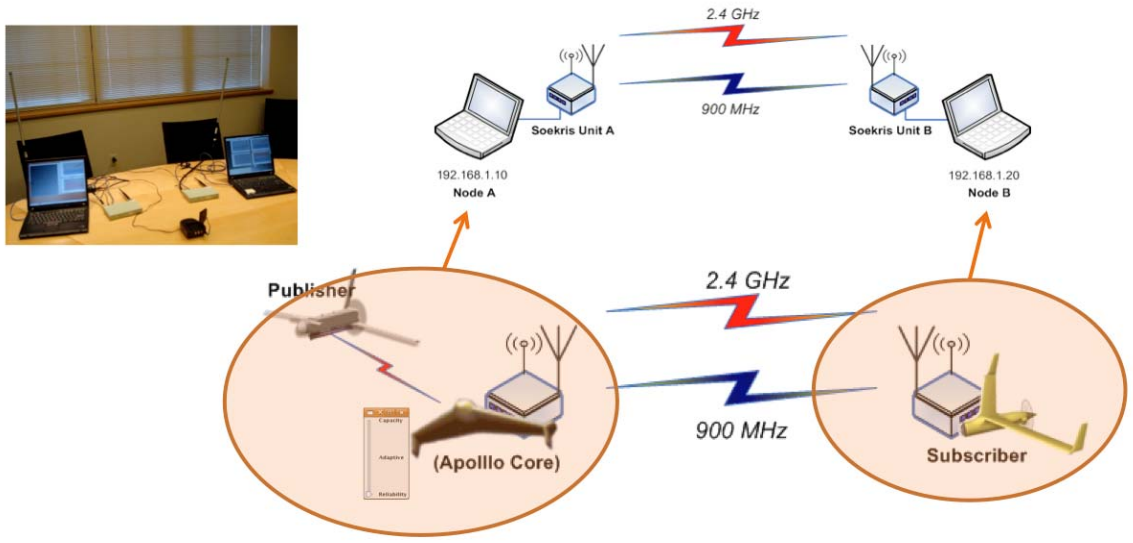


Figure 10. Experimental Setup for Dual-Link Management Controller

In this scenario, a publisher sends information objects to the Apollo Server, which is connected to an edge node through two redundant links (900 MHz and 3.4GHz). The multiple links are supported by different interfaces connected to distinct radios for each link. While the publisher application and Apollo server can choose to bind to any one of the interfaces, the XLayer substrate abstracts both links into a single connection between the two platforms.

The management of both data links is handled by the XLayer to support the application requirements defined by the applications (or by related policies). In this example, the application (represented by the IMS), may define a self-balancing policy for the links (i.e. an adaptive strategy), or strategies that favor robustness or throughput.

In each case, the XLayer will balance the flow across the links provided to satisfy application requirements. Also as part of this demonstration, an RF-flooded was used to compromise one of the links (the 2.4 GHz) and simulated an external attack. The goal was to ensure that, despite of any specific interfaces to which the applications may be bound, the XLayer would transparently balance the flow across the different links to maintain the requirements.

In Figure 11, the demonstration scenario shows the output for each of the three requirement conditions specified by the applications, namely the favoring of capacity, reliability or and adaptive behavior.

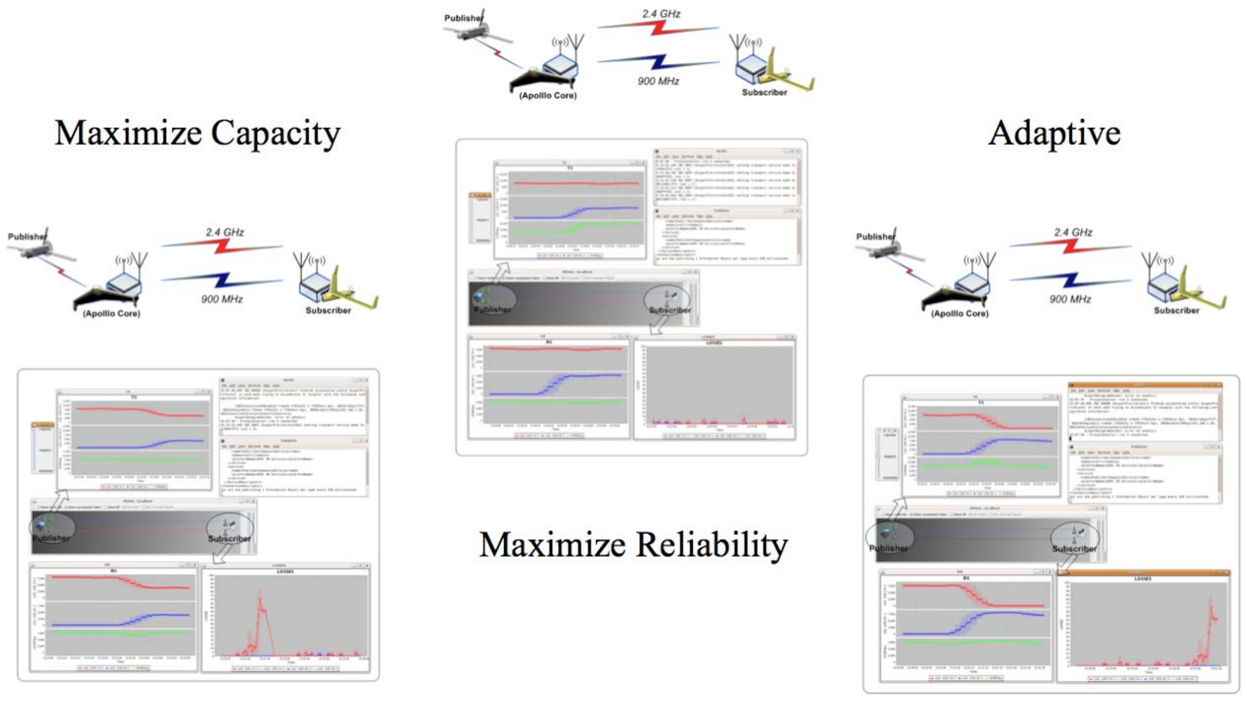


Figure 11. Online Strategies for Dual-Link Management, in Support to Application Requirements

For capacity maximization, the XLayer uses both links as an aggregate pipe, alternating information object transmissions for each of the links, based on their throughput capacity and queue conditions. If favoring reliability (second case), the XLayer would duplicate information management objects on both links, again in accordance with their throughput limits and queue sizes. Information objects sent on both links were matched on at the receiver side and dropped in the case of duplicates. The goal with the duplication of objects was to minimize the loss of objects that could occur due to link failures or external interferences.

In the adaptive scenario, there were no a-priori preferences chosen by the application. The best link from a minimal error rate and maximum throughput perspective, in this case, was chosen for transmission. External effects to the link, such as interference created by the RF-flooder would results in re-allocation of resources to maintain application requirements. Each of the cases illustrated in Figure 11, were demonstrated at the AFRL PI meeting in 2008.

4.3 Dual-Path Algorithms for Tactical Environments

Another capability developed as a controller for the XLayer communications substrate was the dual-path topology control algorithm. This effort was developed in collaboration with Rockwell Collins and evaluated in a simulated environment, and it was published in [1].

The dual-path algorithm bridged information from the transport and physical layers to autonomously create two disjoint communication paths for an end-to-end data stream. The resulting path was dynamically created as a reaction to the data flow and, conditioned to resource availability, ensured that both data paths were node, and link disjoint, eliminating cross-interference.

The goal of the protocol was to enable robust end-to-end data paths, with support from topology control algorithms. While topology control could be obtained through power, frequency or mobility management, our proof of concept implementation [1], focused on mobility management.

The mobility was based on heuristics implemented at each node that would trigger when interfering traffic from the same data flow was detected. Some of the heuristics for the algorithm are illustrated in Figure 12.

Our initial publications of the dual-path algorithm were primarily focused on the description of the algorithm and qualitative analysis of benefits. We refer the reader to [1] for further details and for a description of the pseudo-code used for this controller.

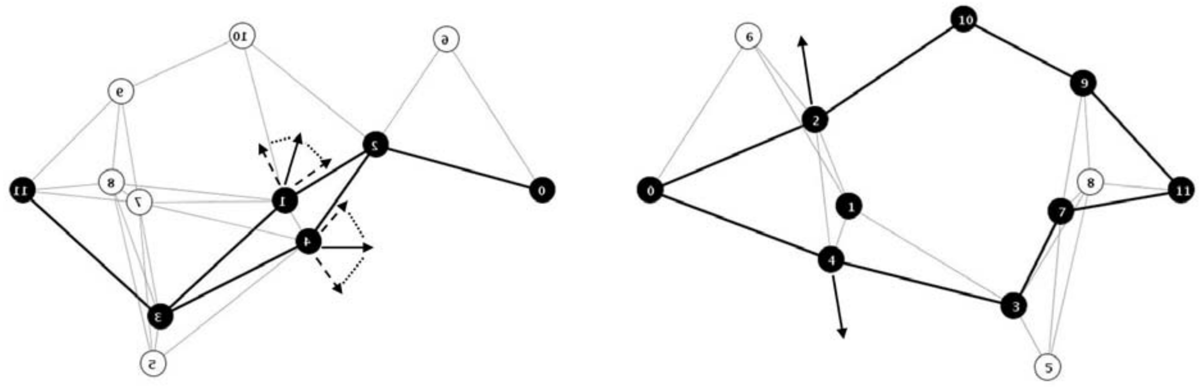


Figure 12. Example heuristics for the dual-path algorithm

4.4 *Adaptive Discovery and Information Dissemination*

One of the core services of XLayer that we developed consisted of a discovery and information dissemination mechanism capable of self-adapting to different network topologies and traffic conditions. The main goal of this mechanism was to reduce the number of retransmissions for broadcast packets while attempting to maintain the same delivery rate of more simplistic broadcast techniques that are known to cause undesired bandwidth congestion.

This adaptive mechanism made use of different broadcasting algorithms that are known to perform better under certain network conditions [20]. Based on the characteristics of the current network topology (i.e. sparse versus dense), the dissemination service would activate a broadcast algorithm that was more suitable for the given network topology and traffic conditions, reducing the number of retransmissions, and improving the effectiveness of the broadcasting algorithm and the overall performance of the dissemination mechanism.

In Figure 13 we show a comparison of a reliable flood mechanism for service discovery based on simple flooding with our self-adaptive approach. The results demonstrate that the delay and coverage (i.e. number of records found) are statistically the same in both cases, but the number of retransmissions is significantly less for the adaptive case.

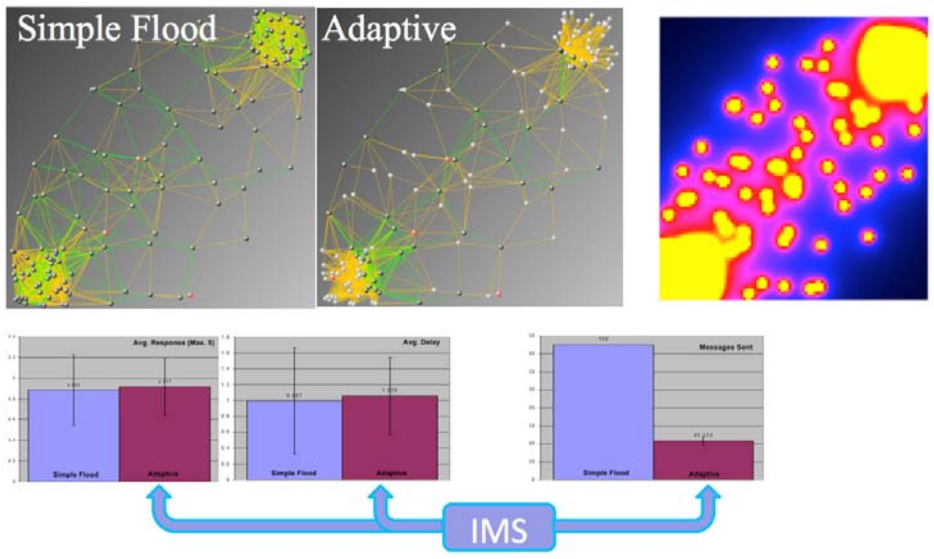


Figure 13. Adaptive Discovery/Dissemination based on Local Network Density

Another approach we explored was the use of dynamic backbones for dissemination of service registrations based on an adaptive CDS-discovery mechanism (Figure 14), providing a mechanism for balancing the trade-off between a higher number of registrations versus higher lookup costs, which could be chosen by the framework on demand.

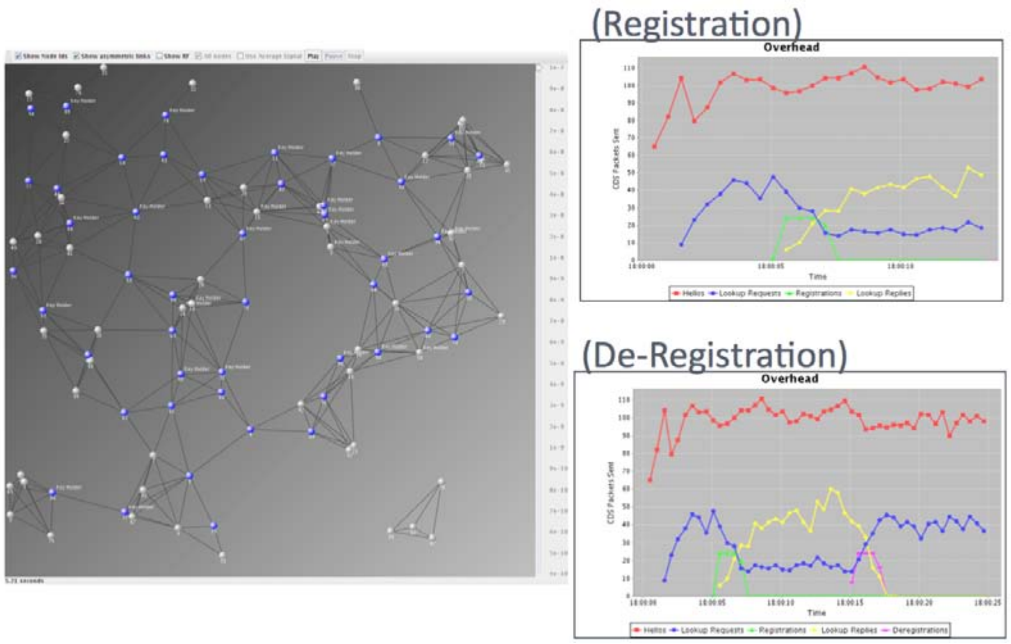


Figure 14. Adaptive CDS-based Discovery

Collectively, these capabilities enabled enhanced discovery and also information dissemination in general. For example, services such as monitoring and link-state routing, which strongly rely on the dissemination of messages across the network, leveraged the capabilities of this self-adaptive dissemination mechanism to efficiently share network and node state information, while reducing the overhead caused by unnecessary retransmissions of messages between nodes.

4.5 Dynamic Gateway Selection

Another capability developed and demonstrated as part of the XLayer communications substrate was the dynamic selection of gateways for cross-domain routing. Cross-domain routing is an important capability in tactical environments. In [8], we have implemented a distributed algorithm for dynamic gateway selection that enables cross-domain routing between different networks running different routing algorithms.

The purpose of this controller is to enable interaction between different networks running different routing algorithms or routing policies coming within contact to one another. The XLayer substrate must be able to detect and recognize both routing algorithms and will determine how to identify and configure gateway nodes to bridge across the networks.

Different routing protocols exist at each node, registered with the XLayer. Only one protocol is active at each network until contact is established. As illustrated in Figure 15, an unknown routing message is checked against all registered protocols and if a match is found a gateway election algorithm is triggered.

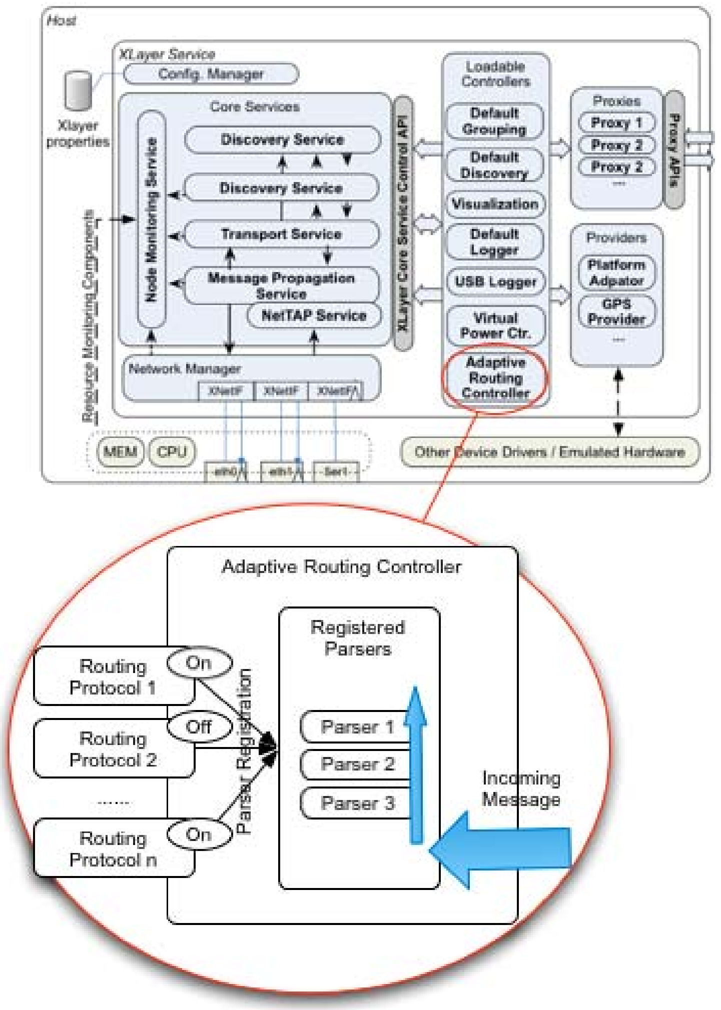


Figure 15. Parsing unknown routing messages

Once elected to become a gateway, a node is responsible for sharing routes across both networks, enabling the seamless routing across both networks. Only gateway nodes execute more than one routing protocol, all other nodes remain on their original protocols, learning new routes to the remote network through the gateway node.

The experimental evaluation of this work was done through simulated scenarios where two networks running the HSLS and OLSR protocols are temporarily within range of one another. A randomly chosen node in one of the networks is continuously trying to send messages to randomly selected receiver on the remote network. Upon connection at the boarder nodes, and gateway election, the sender node finds a route to the receiver and starts sending data. To illustrate the capability, the performance was measured in terms of packet drops and delay in route establishment.

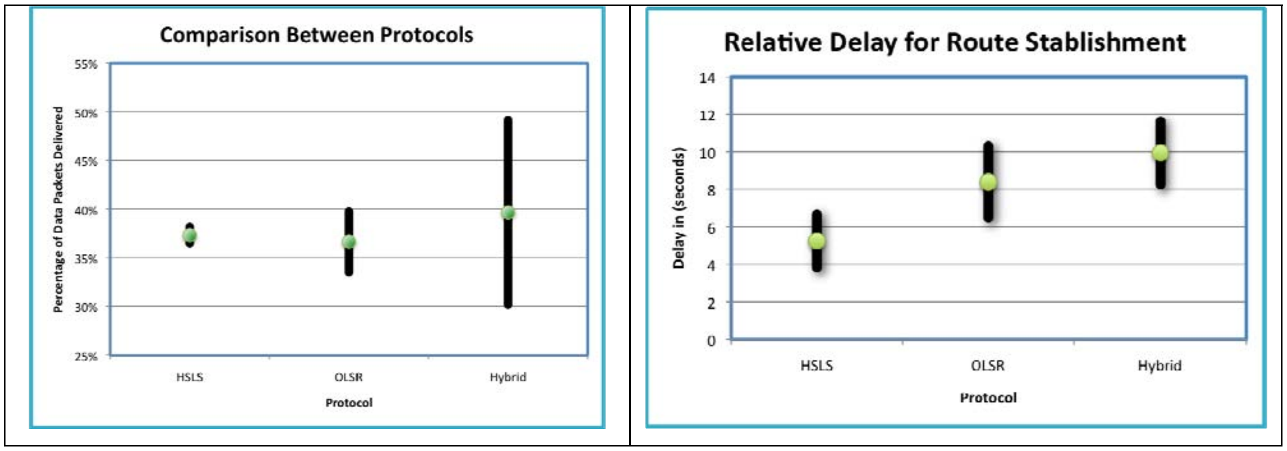


Figure 16. Simulation results for the dynamic gateway controller

The mixed OLSR/HSLS scenario, called hybrid, was compared with a pure OLSR environment where both networks were a-priori running OLSR, and a pure HSLS environment where both networks were a-priori running HSLS (

Figure 16).

4.6 Distributed Monitoring Service

Distributed monitoring was another capability of the XLayer that we developed as one of its core services. XLayer was designed to collect, aggregate, and share a distinctive number of metrics using information available to the node, such as routes, packet drop rates and delay estimations, number of interfaces, CPU usage, memory, disk, network utilization, other predefined metrics, and service or application defined metrics.

Internally, XLayer maintained a table of metrics, each of those identified by a name and a time-series data structure. Every time a new value was provided for a particular metric, the time-series data structure was updated. When the statistics for a metric were requested through the XLayer monitoring API, the time-series was used to compute the average, variance, and trend of all the collected values in the time-series window.

XLayer-aware nodes leveraged on enhanced dissemination mechanisms (see 2.2.2) to share and distribute the information across the network, so that nodes could build a global state of the network that could allow other applications and XLayer services to make informed decisions in order to satisfy particular QoS requirements.

Additionally, we developed a powerful notification mechanism based on simple subscription rules that allowed applications to be notified about metric changes, trends and range violations based on previously specified minimum and maximum thresholds.

4.7 Dynamic Integration of Heterogeneous Networks

The use of controllers to interface with service management systems allows the XLayer substrate to seamlessly monitor and coordinate multiple heterogeneous networks for a common mission. This capability was demonstrated in January 2009 as part of a demonstration scenario at the AFRL.

The test scenario for the proposed demonstration is illustrated in Figure 17 and includes multiple overlapping networks combined through nodes with multiple interfaces. Each network is configured independently at the beginning at the test and the XLayer substrate identifies the configuration and shares state information with other nodes to allow for dynamic resource allocation and reconfiguration as necessary.

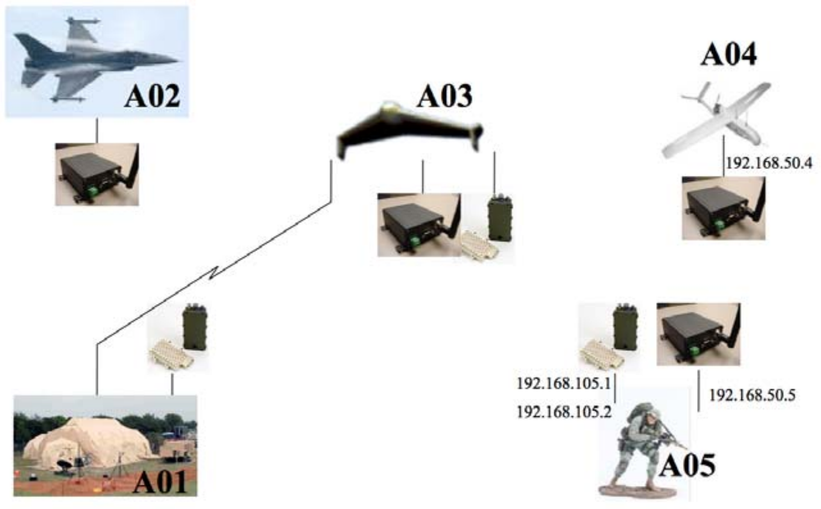


Figure 17. Seamless integration of heterogeneous networks

This part of the work has not been published but it may later be included as part of a joint publication with the AFRL and Raytheon, both collaborators in this demonstration effort.

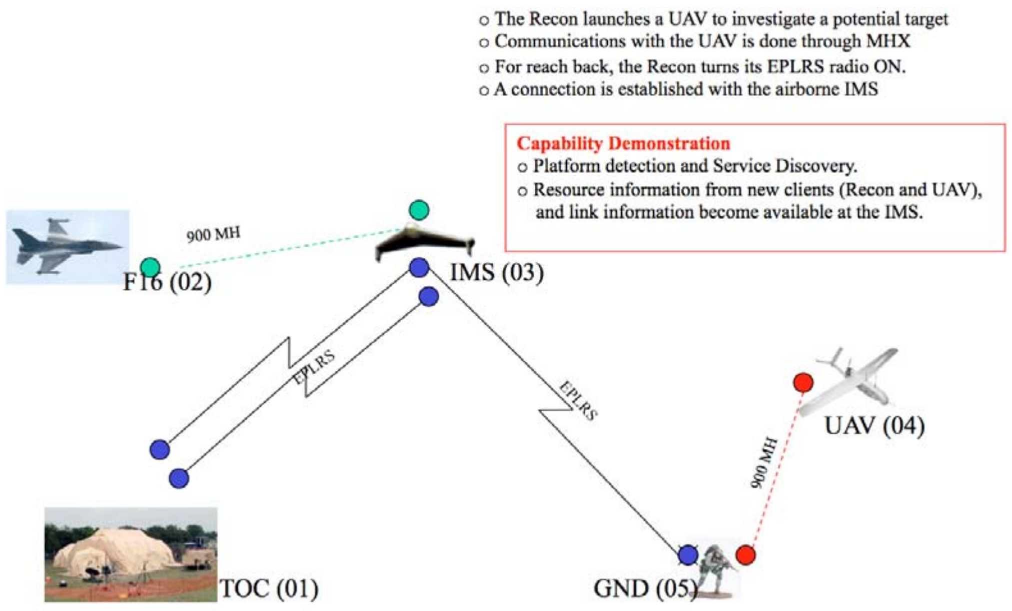

o The Recon launches a UAV to investigate a potential target
o Communications with the UAV is done through MHX
o For reach back, the Recon turns its EPLRS radio ON.
o A connection is established with the airborne IMS
Capability Demonstration
o Platform detection and Service Discovery.
o Resource information from new clients (Recon and UAV), and link information become available at the IMS.
900 MH
F16 (02)
IMS (03)
EPLRS
EPLRS
UAV (04)
900 MH
TOC (01)
GND (05)

Figure 18. Demonstration Phase A – Launching UAV

During the first phase (Phase A) of the demonstration (Figure 18) a Recon ground soldiers launches a small UAV to locate potential threats in the area. The data link maintained between the ground soldier and the UAV is a 900 MHz channel provided by a serial Microhard Radio.

Upon detection of a potential target, the UAV starts publishing images to the IMS. As a subscriber to the images, the TOC starts to receive MIOs from the IMS through the dual link illustrated in Figure 19. For demonstration purposes, the two links between the IMS and the TOC were set with Microlight EPLRS radios, and with an Ethernet connection, representing a high-capacity direct link between the nodes.

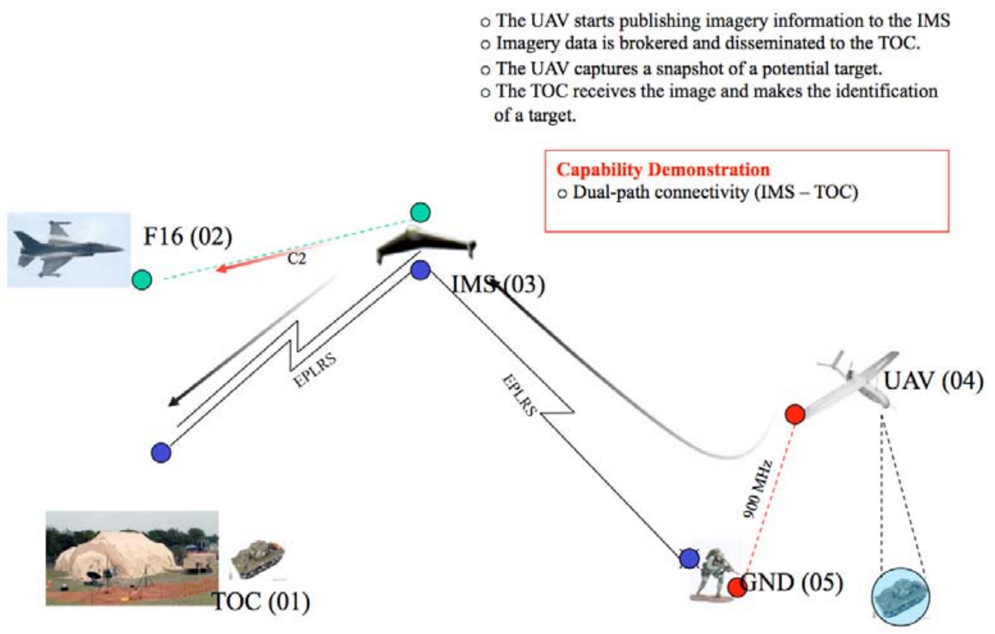

o The UAV starts publishing imagery information to the IMS
o Imagery data is brokered and disseminated to the TOC.
o The UAV captures a snapshot of a potential target.
o The TOC receives the image and makes the identification of a target.
Capability Demonstration
o Dual-path connectivity (IMS – TOC)
F16 (02)
C2
IMS (03)
EPLRS
EPLRS
UAV (04)
900 MHz
GND (05)
TOC (01)

Figure 19. A potential target is located in Phase B, triggering the publications of images from the UAV

The scenario evolves with the TOC tasking the F16 fighter to locate the target (Figure 20). The F16 subscribes to the UAV images and starts receiving the data published by the UAV. During this process, the XLayer is managing the node (and service) discovery, and the resource allocation amongst the different links.

During the subsequent phases of the demonstration, the F16 moves to the location of the target opportunistically managing different data links through its path and seamlessly receiving image updates from the UAV. At all times, the XLayer manages the allocation of the appropriate data-links for message passing, providing seamless discovery and message dissemination across the heterogeneous data links.

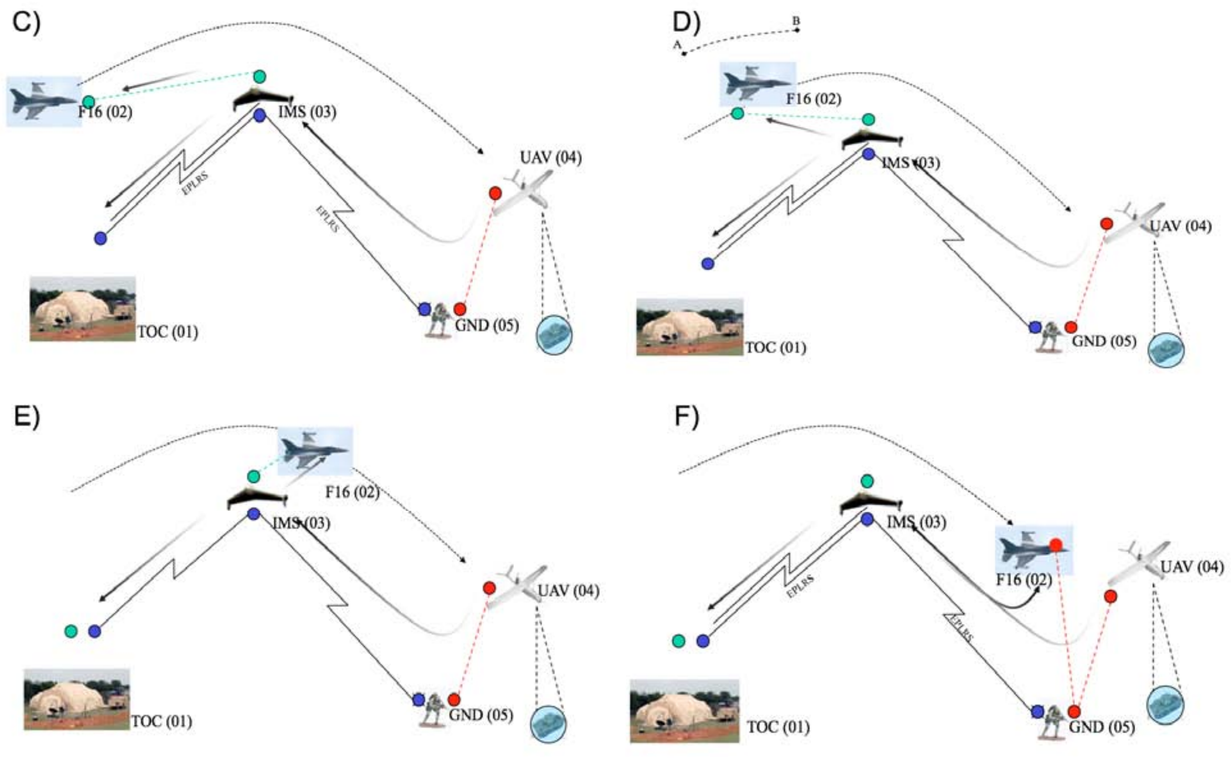


Figure 20. Subsequent Phases of the Demonstration Scenario

5 CONCLUSIONS

The modular architecture of XLayer provides the necessary mechanisms to enable the interaction between applications or middleware and the different levels of the communications substrate, which makes the design of XLayer flexible enough to support a large range of different scenarios and capabilities. This design also facilitates the development of more complex functionality by leveraging on the information and functionality provided by core services that abstract essential capabilities realized across the various layers of the communications stack.

A rich set of APIs allow applications to better adapt and allocate resources for a given task, while at the same time contributing to better define the information and resource needs that can be used by XLayer to allocate the necessary capabilities so as to meet the QoS requirements of applications. Thus, XLayer provides an extensible platform suitable for research and experimentation of new protocols and algorithms for tactical and mobile ad hoc networks for the battlefield.

XLayer was implemented, tested and demonstrated in multiple exercises and experiments carried in collaboration with the Army Research Laboratory, the Air Force Research Laboratory, and commercial companies like Raytheon Co. and Rockwell Collins. During the life of the project, XLayer has been ported to several operating systems such as Linux, OpenWRT, Pyramid, Windows, MacOS X, multiple computing platforms, and tactical radios. Different configurations have been utilized to demonstrate and highlight specific capabilities of XLayer, such as discovery, link sensing, adaptive and predictive routing, topology control and adaptation, enhanced forwarding of broadcast and multicast traffic, and others. When possible, XLayer was executed as part of the communications device (e.g. Linksys-based 803.11 radios, the iPod Touch platforms, and the Soekris-based radios). In other situations, such as the case where XLayer needed to support tactical radios, the communication between the radios and XLayer was realized through the implementation of customized interface adaptors or pseudo-interfaces.

A multi-system demonstration including a tactical Information Management System for an airborne ground-support operation scenario was executed at the Air Force Research Laboratory in early 2009. During such demonstration, XLayer enabled seamless operation across the multiple network and communications environments. The key features of XLayer that made possible such integration include the utilization of multiple redundant links to provide enhanced discovery, node and network resource monitoring, seamless cross-domain routing and on-demand multi-path transport.

XLayer has also been applied in support of key capabilities for tactical Information Management Systems for Airborne Network environments [5][6], quality-of-service enabled data dissemination [13][14][15], emulation environments for airborne networks, and several other applications in security and network management.

6 RECOMMENDATIONS

During the course of this research we have demonstrated several cross-layer algorithms and controllers that can provide a direct improvement on the performance and functionality of tactical information management systems. Furthermore, we have also introduced and demonstrated a modular architecture to support the integration and coordination of multiple controllers. Following our results and conclusions, we recommend the following points for applying or continuing this research effort.

- Additional controllers and cross-layer algorithms: Different applications and operational environments may benefit from different sets of capabilities and possibly additional controllers. During our research, we have identified four topics to illustrate the concept and have developed a few controllers for each topic. New scenarios and applications may require additional controllers for the communications substrate.
- Management Algorithms for the instantiation and orchestration of cross-layer controllers. While we have applied different controllers to different scenarios, a more comprehensive approach capable to opportunistically select, combine and leverage the different is still needed for general applications. Policy-based management infrastructures for QoS support could provide this capability.
- The design of self-organizing coordination controllers is necessary to enable the combination of different capabilities for specific scenarios and applications. There is a need for further research on self-organizing coordination approaches that could benefit from the XLayer framework and the controllers for the different layers.

7 REFERENCES

[1] M. Arguedas, C. Perez, M. Carvalho, A. Granados, K. Hoback, and W. Kraus. Investigating the use of topology adaptation for robust multi-path transport: A preliminary study. In *NCA ‘09: Proceedings of the 8th IEEE International Symposium on Network Computing and Applications*, 2009.

[2] G. Ahn, A. Campbell, A. Veres, and L. Sun. SWAN: Service differentiation in stateless wireless ad hoc networks. In *INFOCOM ’02: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, Volume 2, pp. 457–466, 2002.

[3] V. Bharghavan, K. Lee, S. Lu, S. Ha, J. Li, and D. Dwyer. The TIMELY adaptive resource management architecture. *IEEE Personal Communications Magazine 5*(4), pp. 20–31, 1998.

[4] T. Bova and T. Krivoruchka. Reliable UDP protocol. Internet Draft: http://www.ietf.org/proceedings/99mar/I-D/draft-ietf-sigtran-reliable-udp-00.txt, 1999.

[5] M. Carvalho, A. Granados, W. Naqvi, A. Brothers, J. Hanna, and K. Turck. A cross-layer communications substrate for tactical information management systems. In *MILCOM ’08: Proceedings of the IEEE Military Communications Conference*, pp. 1–7, 2008.

[6] M. Carvalho, A. Uszok, N. Suri, J. Bradshaw, P. Ceccio, J. Hanna, and A. Sinclair. Enabling information management systems in tactical network environments. *Defense Transformation and Net-Centric Systems*, 2009.

[7] M. Carvalho, R. Winkler, C. Perez, J. Kovach, and S. Choy. A cross-layer predictive routing protocol for mobile ad hoc networks. In *Proceedings of the SPIE, 6981. Defense Transformation and Net-Centric Systems*, 2008.

[8] M. Carvalho, C. Perez, and A. Granados. Dynamic gateway selection for cross-domain routing with the XLayer communications substrate. *The International Workshop on Scalable Ad Hoc and Sensor Networks*, Saint Petersburg, Russia, October, 2009.

[9] A. Goldsmith, and S. Wicker. Design challenges for energy-constrained ad hoc wireless networks. *IEEE Wireless Communications Magazine 9*(4), pp. 8–27, 2002.

[10] A. Granados, and M. Carvalho. Building and running the iPod/iPhone Multicast Forwarding Plugin (IMF). Internal Technical Report, April 2009. Available online at https://titan.ihmc.us/svn/manet/trunk/docs/reports/ipod-dev-imf.pdf

[11] A. Granados, and M. Carvalho. Building and running the MOLSR Plugin for the iPod Touch. Internal Technical Report, April 2009. Available online at https://titan.ihmc.us/svn/manet/trunk/docs/reports/ipod-dev-molsr.pdf

[12] S. Kunniyur, and R. Srikant. End-to-end congestion control schemes: Utility functions, random losses and ECN marks. *IEEE/ACM Transactions on Networking (TON) 11*(5), pp. 689–702, 2003.

[13] J. Loyall, M. Carvalho, D. Schmidt, A. Martignoni III, D. Schmidt, A. Sinclair, M. Guillen, J. Edmonson, L. Bunch, and D. Corman. QoS enabled dissemination of Managed Information Objects in a Publish-Subscribe-Query Information Broker. *Defense Transformation and Net-Centric Systems*, 2009.

[14] J. Loyall, M. Gillen, A. Paulos, L. Bunch, M. Carvalho, J. Edmondson, P. Varshneya, D. Schmidt, A. Martignoni III. Dynamic policy-driven Quality of Service in service-oriented systems. *IEEE International Symposium on Object-oriented Real-time Distributed Computing (ISORC)*, Carmona (Parador de Carmona), Spain, May 5-6, 2010.

[15] J. Loyall, A. Sinclair, M. Carvalho, A. Martignoni III, M. Gillen, L. Bunch, M. Marcon. Quality of Service in US Air Force Information Management Systems. *MILCOM*, Boston, MA, October 18-21, 2009.

[16] M. Mirhakkak, N. Schult, and D. Thomson. Dynamic bandwidth management and adaptive applications for a variable bandwidth wireless environment. *IEEE Journal on Selected Areas in Communications 19*(10), pp. 1984–1997, 2001.

[17] P. Sinha, R. Sivakumar, and V. Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. In *INFOCOM '09: Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies, Volume 1*, pp. 202–209, 1999.

[18] R. Sivakumar, B. Das, and V. Bharghavan. Spine routing in ad hoc networks. *ACM Cluster Computing Journal 1*(2), pp. 237–248, 1998.

[19] E. Tromp. The OLSR Multicast Forwarding plugin, 2006. Available online at http://olsr.bmf.sourceforge.net

[20] B. Williams, and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *MOBIHOC '02: Proceedings of the 3rd ACM International Symposium,* 2002.

SYMBOLS, ABBREVIATIONS, AND ACRONYMS

API	application programming interface
BMF	basic multicast forwarding
CEDAR	core-extraction distributed ad-hoc routing
CDS	connected dominating set
CPU	central processing unit
dRSVP	dynamic resource reservation protocol
ECN	explicit congestion notification
EPLRS	enhanced position location reporting system
HSLS	hazy-sighted link state routing
IMS	information management system
JNI	Java native interface
MAC	medium access control
MANET	mobile ad-hoc network
MIO	management information object
MLAB	an emulation testbed for mobile ad-hoc networks
OLSR	optimized link state routing
POLSR	predictive optimized link state routing
QoS	quality of service
RSVP	resource reservation protocol
SWAN	self-organizing wireless adaptive network
TCP	transmission control protocol
TIMELY	adaptive resource management architecture for resource reservation
TOC	tactical operations command
TUN/TAP	virtual network kernel driver for software network interfaces
UAV	unmanned aerial vehicle
UDP	user datagram protocol